TEXT CATEGORIZATION USING ONLY FRAGMENTS OF DOCUMENTS

PILÁSZY, ISTVÁN – DOBROWIECKI, TADEUSZ

Key words: machine learning, text categorization, classifier ensembles.

CONCLUSIONS

In this paper we presented a lot of experiments that examine how the particular parts of the documents do contribute to the performance of a classifier. We evaluated text classifiers on two very different text corpora. We conclude that some parts of the text are more important from the point of text classification performance. Giving higher weights to more important parts can increase the performance of the classifier. The question, that which parts are more or less important depends on the nature of the documents in the corpora. Some tasks that remains to be done:

More text corpora should be investigated.

In section 6.4 we optimized the number of features to be kept independent from the section. However, it could be optimized for each section.

Splitting the documents into parts of 50 words, to examine what if the parts are of equal size not only inside a document, but among the documents too.

When splitting documents into *k* equal parts, we may combine the classifiers resulted from different *k* values.

ABSTRACT

The aim of Text Categorization (TC) is to automatically assign documents to a set of predefined categories. TC systems use machine learning (ML) to automatically build classifiers based on a set of labeled documents. TC systems usually use the so-called bag of words approach to represent documents suitable for machine learning, i.e. for each document, for each word only the number of occurrences counts, the order of words is completely disregarded. In this paper we present experiments leading to the conclusion, that some parts of the texts are more important than the others, at least from the point of TC.

INTRODUCTION

The task of text categorization is to automatically assign labels to documents. Labels are drawn from a predefined set (often called categories), which means that TC systems cannot invent new categories. TC methods are used to filter out spam, to find interesting information on the web, to classify news stories, to build web directories etc. With the growth of internet the importance of TC increases. Until the late '80s the most popular approach was to build classifiers manually, i.e. by defining a set of rules encoding an expert's knowledge (1). Nowadays the best TC systems use supervised machine learning to derive these rules, often outperforming manually created ones, and saving the human effort. To automatically build a classifier, one needs a set of labeled documents, the so-called training set. To evaluate the classifier, it must be applied on another set of labeled documents, the

so-called testing set. The comparation of the expected and predicted labels provides an insight into the performance of the classifier.

TRADITIONAL TC SYSTEMS

Text categorization consists of text pre-processing, transformation into a vector space, using machine learning on training documents to obtain a model (learning phase), applying that model to the test documents and finally comparing the expected and predicted labels (testing phase). In the pre-processing phase, stop words are often removed. Stop words are the most common words without a strict problem related meaning, e.g. *a, an, to, and, or, but,* etc. These words do not contribute to the recognition of the label of the document. The removal of them only slightly degrades or even increases the performance, and speeds up the learning and testing phase: a great percentage of the words can be considered stop words. Word stemming and lowercase conversion is common during preprocessing. The aim of these procedures is to reduce the dictionary. When only few training documents are available, these methods increase the performance, but when the opposite holds, they can decrease it. The majority of the learning methods learn an input-output mapping from a real vector space to a binary value (true or false, +1 or -1, spam or not spam). For this reason, documents have to be transformed into this space. The most commonly used transformation is the so-called tf-idf term weighting scheme (2). Roughly speaking, terms are words, but a term is a more general concept, it may mean phrases, etc. In tf-idf term-weighting scheme documents are represented as vectors, each dimension corresponds to a term. Document vectors (columns) form together the so-called term-document matrix *TD*:

$$TD(t,d) = TF(t,d) \cdot IDF(t),\ IDF(t) = \log \frac{N}{DF(t)},\ DF(t) = \sum_{d:TF(t,d)\neq 0} 1,\ N = \sum_{d} 1 \qquad (1)$$

where *TF(t, d)* is the number of occurrences of the *t*-th term in the *d*-th document, *DF* is so-called document-frequency, *IDF* is the inverse document-frequency. This formula expresses the idea that the words that frequently occur in one document are important, but the more there are documents they occur in, the less important the words are. The term-document matrix can be quite large. To reduce its size, term-selection (feature selection) methods are used. The most commonly used approach is to select only *n* terms with the highest *DF* value. If more terms do exist with the same *DF* value that are to be deleted, some care has to be taken to implement feature selection deterministically, for the sake of comparable results. Documents may have different length, which may results in anomalies. To avoid this, Euclidean normalization is applied to the columns of the *TD* matrix. There is no proof that such normalization is the best, but this is the most prevalent operation (β=2):

$$TD'(t,d) = \frac{TD(t,d)}{\sqrt[\beta]{\sum_i TD(t,d)^\beta}} \qquad (2)$$

After this transformation we are already able to apply machine learning algorithms. A great deal of ML methods exists which has been applied to TC. The most popular supervised machine learning framework in the TC community is the so-called support vector machines (SVM). It yields a linear model, i.e. it creates a function that linearly combines the features. That function defines a hyperplane that separates positive examples from negative ones. There are many other algorithms that also create a linear model (Naïve Bayes, perceptron, logistic regression). Positive and negative discrimination is based on the sign of the

separating function. In case of more than two categories, for each category we separately train it against the others. This technique is suitable to classify documents belonging to more than one category. If we know that each document belongs to one category only, we choose the category that yields the highest score.

SUPPORT VECTOR MACHINES

SVM tries to find a hyperplane that separates positive and negative examples with large margin and low training error (5). In the simplest case no training arrow is allowed and the margin is the distance between the decision surface and the training example closest to the surface. The margin defines two more hyperplanes, positioned at a distance of margin from the decision surface. Training error occurs if we allow examples to be on the wrong side of these hyperplanes, which is allowed by a more general SVM. We can increase the margin at the expense of increasing training error. SVM minimizes a fixed linear combination of the reciprocal of the margin and the training error. There is a non-linear extension to the SVMs that exploits an interesting property: SVM does not use examples as they are, but as the inner product with other examples. The non-linear extension replaces the inner product with a more general kernel-function. The kernel function must satisfy the property that there exists a function $\mathbf{\Phi}$: $\mathbf{R}^n \rightarrow \mathbf{R}^m$, such that:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}_j) \quad (3)$$

The model that SVM does create is in the form: (8)

$$f(\mathbf{x}) = \left(\sum_i \alpha_i y_i \mathbf{\Phi}(\mathbf{s}_i)^T\right) \cdot \mathbf{\Phi}(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b \quad (4)$$

Where $\mathbf{x}$ is the test example, the sign of f($\mathbf{x}$) indicates the category of $\mathbf{x}$, $\mathbf{s}_i$ are the so-called support-vectors (a subset of training examples), y_i is the category of $\mathbf{s}_i$ (-1 for negative, 1 for positive documents), K is the kernel function, α_i and b are the parameters of the model, chosen to maximize the margin. The normal of maximal-margin hyperplane is in the span of $\mathbf{\Phi}(\mathbf{s}_i)$ (8).

EVALUATION OF TEXT CLASSIFIERS

TC systems may of course make mistakes and we need performance measures to compare them. For a single category c, the following measures are commonly used [1]:

precision: $$p = \frac{TP}{TP + FP} \quad (5)$$

recall: $$r = \frac{TP}{TP + FN} \quad (6)$$

F_1: $$\frac{2 \cdot p \cdot r}{p + r} \quad (7)$$

where TP stands for true positives (the example was preclassified as positive and the classifier classifies it is positive), FP for false positives (the example was negative and the classifier classifies it is positive), FN for false negatives (the example was positive and the classifier classifies it is negative).

In case of multiple categories c_i the following averages are commonly used (1):

micro-precision: $$Mip = \frac{\sum_i TP_i}{\sum_i TPi + FPi},$$

macro-precision: $$Map = \frac{\sum_i p_i}{\sum_i 1} \quad (8)$$

micro-recall: $$Mip = \frac{\sum_i TP_i}{\sum_i TPi + FNi},$$

macro-recall: $$Mar = \frac{\sum_i r_i}{\sum_i 1} \quad (9)$$

micro-F_1: $$MiF_1 = \frac{2Mip \cdot Mir}{Mip + Mir},$$

macro-F_1: $$MaF_1 = \frac{2Map \cdot Mar}{Map + Mar} \quad (10)$$

PREPARATION OF THE EXPERIMENTS

To build TC systems, we need a set of labeled training examples. Then to evaluate a TC system, we need another set of labeled documents, the testing set. However, in practice this set of documents are unlabeled, that is why we apply TC (and use all labeled examples for training). We use two very different text corpora for the experiments: music news from music.hu, and scientific articles from Elsevier's ScienceDirect® (7). The corpus is built from articles from music.hu Hungarian music portal and contains 1482 training documents and 585 test documents, each assigned to one of 8 categories. Categorization is based on the music genre of an article. Articles are not structured, only the title and the document body can be identified. Articles are short, on the average 1225 characters or 201 words per document. The other corpus was built from Elsevier's ScienceDirect® website. We have downloaded 1970 articles from 3 different journals: 1000 from *Computer Networks*, 570 from *International Journal of Medical Informatics*, 400 from *Cell Biology International*. We assigned the first 1379 article to the training set, and the rest 591 articles to the test set. Categorization is based on the journal name of the article, thus we have 3 categories altogether. Documents have common structure: *title*, *keywords*, *abstract*, *authors*, *references* sections are common in almost each document. Each of the downloaded documents possesses all of these sections. Other kinds of sections are considered „*rest*". Articles are long, on the average 30992 characters or 5819 words per document. In each corpus the training – test splitting is based on the creation time of the documents, i.e. all the test documents are newer than any of the training documents. In TC literature the popular way is to randomly split them, which yields better results. However, we do not agree with this approach, because in practice the categories of the test documents are unknown, and when we are given a new document, most likely newer than any other having been seen before, then we expect the TC system to do the best in this case. For SVM learning, we have used the SVM^{light} open-source software (3, 4).

EXPERIMENTS AND RESULTS

Basic experiments

To make the results comparable, we will evaluate a typical TC system on these corpora, with multiple parameter-assignments. The adjustable parameters are:

β in the normalization in the eq. (2)

The number of terms (features) to be kept after feature selection: *n*

Type of the kernel: linear ($K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_j$) or second order polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_j + 1\right)^2$ (11)

The obtained results are in the Table 1. For the sake of simplicity, in the followings we will use the same parameter settings for each experiment if not noted otherwise, i.e.: *n*=12000, β=1, polynomial kernel, because the sum of percentages is the highest in this column of the table.

Table 1

Results of basic experiments

	n=12000				n=6000			
	β=1		β=2		β=1		β=2	
	poly	linear	poly	linear	poly	linear	poly	linear
music.hu MiF1	0,699065	0,698975	0,654886	0,685094	0,677596	0,676979	0,641471	0,666025
music.hu MaF1	0,503946	0,503461	0,427004	0,468906	0,512042	0,51104	0,415971	0,517704
Elsevier MiF1	0,987277	0,987277	0,988095	0,987256	0,991525	0,991525	0,988955	0,99067
Elsevier MaF1	0,986583	0,986583	0,98795	0,986952	0,991488	0,991488	0,988104	0,990088

Distinguishing the title words

We examined what will happen, if we would index the document title and the body separately, by prefixing the words with „t_" or „b_", indicating whether they come from the title or the body. It seems a reasonable assumption, that words from the title are more informative than the words from the body, thus we should give them more weight. We modified the TF function in the eq. (1), multiplying it by a constant *tfw*. The title words occur more rarely than the body words, so feature selection throws out most of them. To keep these important words, we multiply their *DF* function with a constant *dfw*, but only in the feature selection phase. We have tried out multiple *dfw* and *tfw* values. The MaF_1 results are presented in the Table 2. It seems that giving more weight to the title than to the body can improve classification performance. Keeping only the title (when *dfw* and *tfw* tend to infinity) is better than keeping only the body (*dfw* and *tfw* are zero) in the music.hu corpus, but the opposite is true for the other corpus. When *dfw* are *tfw* are very different, the classification performance decreases.

Table 2

MaF_1 results of distinguishing title (in %)

on music.hu corpus

dfw \ tfw	0	1e-6	1	2	3	5	10	1e6
0	46.9	46.9	46.9	46.9	46.9	46.9	46.9	46.9
1e-6	46.9	46.9	46.9	46.9	46.9	46.9	46.9	46.9
1	46.9	46.5	48.0	50.0	54.8	56.0	55.4	46.2
2	46.9	46.0	47.9	50.4	54.7	56.0	55.4	46.0
3	46.9	45.7	48.3	50.1	54.6	58.5	56.5	50.4
5	46.9	46.6	48.7	52.3	56.0	58.5	56.5	50.4
10	46.9	46.6	48.7	52.0	56.0	58.5	56.5	50.4
1e6	46.9	46.6	48.7	52.3	56.0	58.5	56.5	50.4

on elsevier corpus

dfw \ tfw	0	1e-6	1	2	3	5	10	1e6
0	46.9	46.9	46.9	46.9	46.9	46.9	46.9	46.9
1e-6	46.9	46.9	46.9	46.9	46.9	46.9	46.9	46.9
1	46.9	46.5	48.0	50.0	54.8	56.0	55.4	46.2
2	46.9	46.0	47.9	50.4	54.7	56.0	55.4	46.0
3	46.9	45.7	48.3	50.1	54.6	58.5	56.5	50.4
5	46.9	46.6	48.7	52.3	56.0	58.5	56.5	50.4
10	46.9	46.6	48.7	52.0	56.0	58.5	56.5	50.4
1e6	46.9	46.6	48.7	52.3	56.0	58.5	56.5	50.4

Switching on and off some parts of the documents

Documents in the Elsevier corpus have common structure: documents consist of 6 parts: *title*, *keywords*, *abstract*, *authors*, *references*, and the *rest*. In the previous experiment we considered only the title and the body, but with various weights. In this experiment we consider only zero or one for *dfw* and *tfw*, but now we define separate *dfw* and *tfw* for each of the 6 part of the document. This means 2^6=64 experiments. The results are presented in the Table 3. Please note that all the best results for MiF_1 and MaF_1 retain the *abstract* and the *authors* sections. It is interesting to compare the cases when only one part is kept: the most informative part is the *abstract* (98.8%), then comes the *rest* (98.8%) (i.e. the main body of the scientific article), the *references* (98.6), the *keywords* (95.9), the *title* (93.9), and the *authors* (92.2). Note how good is the *references* section, and the *title* is a wee bit better than the *authors*. In case of MaF_1 the order does not change.

Table 3

MiF_1 and MaF_1 when keeping only parts of the documents (in %)
tka stands for Title, Keywords, Abstract; arr stands for Authors, References, Rest

MiF_1 (%)

tka \ arr	000	001	011	010	110	111	101	100
000		98.8	98.8	98.6	99.0	98.7	98.7	92.2
001	98.8	98.8	98.8	99.5	99.5	98.9	98.9	99.2
011	99.0	98.8	98.8	99.6	99.6	98.9	98.9	99.2
010	95.9	98.7	98.7	98.6	99.0	98.7	98.7	97.1
110	97.3	98.7	98.8	98.5	99.0	98.7	98.7	98.0
111	99.0	98.8	98.8	99.6	99.6	98.9	98.8	99.0
101	98.9	98.8	98.8	99.6	99.5	98.9	98.9	99.2
100	93.9	98.7	98.8	98.6	99.1	98.7	98.7	96.8

MaF_1 (%)

tka \ arr	000	001	011	010	110	111	101	100
000		98.8	98.8	98.6	99.0	98.7	98.7	91.0
001	98.6	98.8	98.8	99.4	99.4	98.9	98.9	99.1
011	98.8	98.8	98.8	99.5	99.5	98.9	98.9	99.0
010	96.1	98.7	98.7	98.6	99.0	98.7	98.7	96.8
110	97.5	98.7	98.8	98.5	99.0	98.7	98.7	97.9
111	98.8	98.8	98.8	99.5	99.5	98.9	98.8	98.8
101	98.7	98.8	98.8	99.5	99.4	98.9	98.9	99.0
100	93.6	98.7	98.8	98.6	99.1	98.7	98.7	96.6

Creating separate examples for each section

In the previous experiments each document corresponded to one example, only the words were changed with a prefixed string (e.g. „t_" indicated a *title* word). For the music.hu corpus, now we will examine, what happens, if we train a model that uses only the title of the documents, and another model that uses only the body of the documents, and then we will combine the output of the classifiers. For the Elsevier corpus we create thus six models per category, each corresponding to one section of the articles. To obtain the best results, we re-examined feature selection. Besides n (the number of features to be kept) we introduced another parameter m, the minimal number of *DF* for a feature to be kept. This is important in small sections, where the limit of n is far beyond the number of the features. We have found that for music.hu, the best values are n=12000, m=1, and for the Elsevier corpora, it is n=12000 and m=7. The combination of the classifier outputs is done by summarizing the scores of classifiers for each section. To obtain the best combined results, we choose that category for the test document that achieved the highest score. The results are presented in the Table 4. Please note that the combined results are always better than the result for any section. It is interesting to see the order of the sections in the Elsevier corpus: the *rest* section yields the best results, then immediately follows the *references*, then the *abstract*, the *keywords*, the *authors* and the *title*.

Table 4

Results of separately training for each section of the documents and then combining the classifiers (in %)

corpus	document section	MiF1	MaF1
music.hu	title	56,80	45,07
	body	66,98	46,46
	combined	68,03	55,90
elsevier	title	84,29	83,48
	keywords	87,78	87,31
	abstract	97,37	97,00
	authors	87,68	86,62
	references	98,39	98,58
	rest	98,81	98,80
	combined	99,83	99,85

Equally splitting up the documents

In the previous section we split up the documents at the section boundaries. Sections are of unequal size, which may cause anomalies. Now we split up documents into equal parts. These fragments are of equal size in one document, however not between documents. Let k denote the number of such parts. We may expect that some parts, e.g. the first and the last part of the documents, are more important than the others from the point of categorization performance. We evaluated classifiers with different values for k, from 1 to 10. For a fixed k, we have k distinct classifiers, the first trained on the first part of the documents, ..., then the k-th trained on the k-th (last) part of the documents. We have applied the same combination method as in the previous experiment, and used the same values for n and m. MiF_1 and MaF_1 values for the music.hu and elsevier corpus, for different k values, are in the Table 5.a, 5.b, 5.c, 5.d. All the results for different k values in the tables are averaged and combined into one diagram, presented next to the tables. More precisely, these figures present the average of the following functions:

$$f_{k,l}(x) = \begin{cases} NaN & \text{if } x < (l-1)/k \text{ or } x \geq l/k \\ F_1(k,l) & \text{otherwise} \end{cases}$$

$$\text{where } k \in \{1,\ldots,10\},\ l \in \{1,\ldots,k\}, \qquad (12)$$

where $F_1(k,l)$ is the MiF_1 or MaF_1 for the l-th part of the document when being split into k parts, and NaN stands for „Not a Number" (it does not influences the average in that point). Please note than in most rows the combined results are better than any result in the particular row. We find it very interesting also to compare the shape of the diagrams. It suggests, that in the music.hu corpus most of the information is contained in the head of the documents, while in the Elsevier corpus it is contained in the head and the end of the documents (order of sections: *title*, *keywords*, *abstract*, *authors*, *rest*, *references*).

Table 5.a

MiF_1 values for the music.hu corpus (%)

		which part									
k	combined	1	2	3	4	5	6	7	8	9	10
1	70.8	70.8									
2	66.2	65.3	58.3								
3	61.5	61.7	49.7	49.5							
4	57.1	57.5	47.3	42.3	44.1						
5	54.5	55.8	43.7	42.9	38.6	38.2					
6	52.8	50.4	43.2	41.3	37.8	37.3	35.4				
7	49.7	48.9	43.7	40.0	37.1	37.8	36.0	34.3			
8	49.7	46.9	40.8	37.5	37.2	35.8	32.8	33.3	33.5		
9	45.5	46.4	32.9	36.5	38.8	31.2	35.8	34.7	35.0	33.4	
10	48.3	34.5	44.3	32.1	35.7	37.3	33.2	34.1	35.6	33.3	30.4

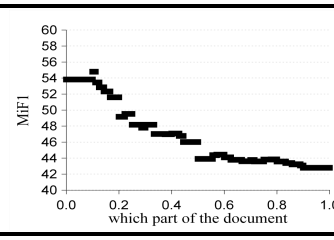


Table 5.b

MaF_1 values for the music.hu corpus (%)

		which part									
k	combined	1	2	3	4	5	6	7	8	9	10
1	57.0	57.0									
2	50.7	52.2	41.7								
3	43.2	44.5	30.8	30.3							
4	39.1	39.5	29.1	21.4	24.3						
5	30.9	41.1	21.6	21.8	19.8	18.9					
6	34.0	35.6	19.7	26.5	21.5	17.6	18.5				
7	32.8	36.8	24.6	19.1	17.5	19.4	14.6	19.3			
8	30.7	36.2	23.6	18.1	20.7	15.4	15.3	15.8	17.2		
9	20.2	31.8	16.5	18.6	17.5	13.4	16.7	21.5	16.7	15.8	
10	23.4	18.5	27.1	16.3	17.1	18.2	17.8	18.1	15.4	16.8	17.7

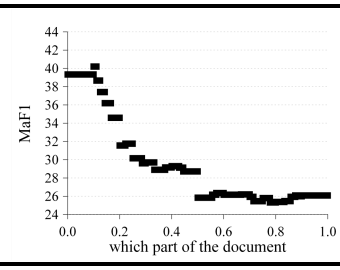


Table 5.c

MiF_1 values for the Elsevier corpus (%)

		which part									
k	combined	1	2	3	4	5	6	7	8	9	10
1	98.6	98.6									
2	99.2	99.2	98.6								
3	99.2	99.3	97.8	98.8							
4	99.2	99.2	97.8	97.0	99.0						
5	99.3	99.3	97.6	97.3	97.3	99.2					
6	99.3	99.0	97.3	96.6	96.4	97.6	99.2				
7	99.2	99.2	97.6	96.8	94.9	96.1	97.0	99.0			
8	99.2	99.0	97.8	95.9	95.9	95.3	96.1	97.1	98.6		
9	99.3	99.0	97.1	95.9	94.9	94.2	94.8	96.1	98.1	98.6	
10	99.3	98.1	98.5	97.1	95.9	94.9	94.2	95.8	95.1	96.3	97.0

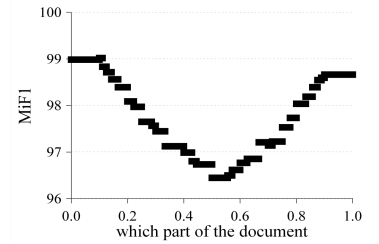
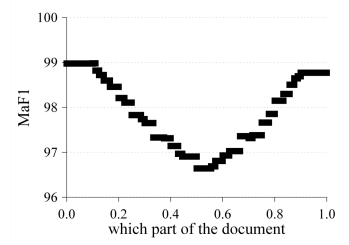


Table 5.d

MaF_1 values for the Elsevier corpus (%)

		which part									
k	combined	1	2	3	4	5	6	7	8	9	10
1	98.7	98.7									
2	99.2	99.2	98.7								
3	99.2	99.4	98.0	98.8							
4	99.2	99.2	98.0	97.3	99.1						
5	99.4	99.2	97.9	97.4	97.3	99.2					
6	99.4	98.9	97.5	96.9	96.7	97.7	99.2				
7	99.2	99.1	97.8	96.9	95.2	96.4	97.0	99.1			
8	99.2	98.9	97.9	96.3	96.2	95.4	96.4	97.4	98.8		
9	99.4	98.9	97.3	96.3	95.1	94.5	95.0	96.4	98.3	98.8	
10	99.4	98.3	98.4	97.2	96.3	95.1	94.5	95.8	95.4	96.5	97.3



REFERENCES

(1) Fabrizio Sebastiani: Machine learning in automated text categorization, ACM Computing Surveys (CSUR), Vol.34 Issue 1, 2002., ACM Press, New York, NY, USA, pp. 1–47. – (2) A. Aizawa: An information-theoretic perspective of tf-idf measures, Information Processing and Management: an International Journal archive, Vol. 39, Issue 1, 2003., pp. 45–65. – (3) Thorsten Joachims: Text categorization with support vector machines: learning with many relevant features, Proc. of ECML-98, 10th European Conference on Machine Learning, Springer Verlag, Heidelberg, DE, 1998., pp. 137–142. – (4) T. Joachims: Making Large-Scale SVM Learning Practical. In: Advances in Kernel Methods - Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola (ed.), MIT Press, 1999. – (5) Vladimir N. Vapnik: The Nature of Statistical Learning Theory. Springer, New York, 1995. – (6) David D. Lewis: Evaluating Text Categorization, Proc. of Speech and Natural Language Workshop, pp.312-318, 1991. – (7) Elsevier's ScienceDirect®: http://www.sciencedirect.com/ – (8) C.J.C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, Vol. 2, Number 2, p. 121-167.